Tennis Ball

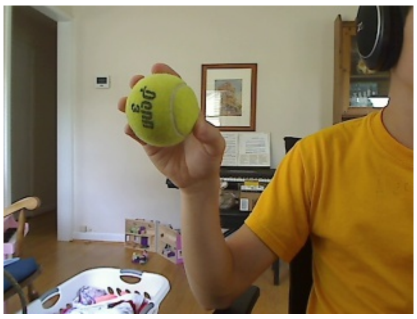
This tutorial proposes options to find a tennis ball in a picture using two different techniques. This will also play with parameters, to show how to avoid finding false positive, i.e objects that are nowhere near the one we are looking for.

First, let's load the image that will be the base for our image finding exercice.

```
(ns divine-briars
  (:require
    [opencv3.utils :as u]
    [opencv3.colors.rgb :as color]
    [opencv3.core :refer :all]))
```

```
nil
```

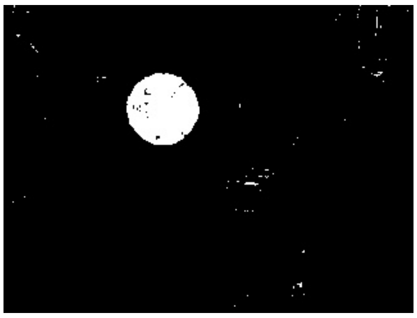
```
(def img (-> "http://i.imgur.com/uoNRu60.jpg" (u/mat-from-url) (u/resize-by 0.5)))
(u/mat-view img)
```



Using Hough Circles

```
(def hsv (-> img clone (cvt-color! COLOR_RGB2HSV)))
(def thresh-image (new-mat))
(in-range hsv (new-scalar 50 100 0) (new-scalar 95 255 255) thresh-image)

; the ball is here
(u/mat-view thresh-image)
```



```
(let[ circles (new-mat) output (clone img) minRadius 25 maxRadius 40 ]

(hough-circles thresh-image circles CV_HOUGH_GRADIENT 1 minRadius 120 15 minRadius
maxRadius)

(dotimes [i (.cols circles)]
  (let [ circle (.get circles 0 i) x (nth circle 0) y (nth circle 1) r (nth circle 2)  p
(new-point x y)]
  (opencv3.core/circle output p (int r) color/plum 2)))

; the ball is detected
(u/mat-view output))
```

Method2: Using Find Contours

Find object with find contours gives less chance to parameters, the **find-contours** function from core does pretty much all for us.

```
(let[ hsv (-> img clone (cvt-color! COLOR_RGB2HSV))
      thresh-image (new-mat)
      contours (new-arraylist)
      output (clone img)]

(in-range hsv
          (new-scalar 50 100 0)
          (new-scalar 95 255 255)
          thresh-image)

(find-contours
  thresh-image
  contours
  (new-mat)  ; mask
  RETR_LIST
  CHAIN_APPROX_SIMPLE)

(dotimes [ci (.size contours)]
 (if (> (contour-area (.get contours ci)) 100 )
  (draw-contours output contours ci color/plum FILLED)))

(u/mat-view output))
```